Trifecta: Faster High-Throughput Three-Party Computation over WAN Using Multi-Fan-In Logic Gates

Sina Faraji
University of Waterloo
sina.faraji@uwaterloo.ca

Florian Kerschbaum
University of Waterloo
florian.kerschbaum@uwaterloo.ca

ABSTRACT

Multi-party computation (MPC) has been a very active area of research, and recent industrial deployments exist. Practical MPC is currently limited to low-latency, high-throughput network setups, i.e., local-area networks (LAN). However, many use cases require the participation of different entities located in different data centers, i.e., communication over wide-area networks (WAN). Although, constant-round MPC exists, it has very high communication cost. In this paper we investigate the reduction of the round complexity of secret-shared based multi-party computation. We propose a new three-party computation protocol that allows to compute multi-fan-in gates in one round without any precomputation. Our protocol outperforms related work, including constant-round protocols, over WANs. For example, we improve throughput of AES-128 over WAN by a factor of more than 2.2× compared to related work.

KEYWORDS

Secure Multi-Party Computation,Binary Circuits,Multi-Fan-In Gates

1 INTRODUCTION

Secure (multi-party) computations (MPC) enable the privacy-preserving processing of joint data. In an MPC, two or more parties jointly compute a function with the condition that other than the output, no party learns anything about the input of other parties. In recent years, many theoretical MPC constructions have found applications in practice. For example, Google and Mastercard have deployed a two-party computation for ad conversions [28, 51] and Meta has released the CrypTen toolkit for machine learning in multi-party computations [32].

There are two fundamental approaches to secure multi-party computation. In the first approach, initially proposed by Yao in his seminal work [49], the parties use a garbled Boolean circuit to evaluate the function in a single round of communication. In the second, the parties secret-share their inputs among each other and compute the circuit in multiple rounds. Although constant round protocols like the Yao's protocol [49] for 2-party and Beaver et al.'s protocol [4] for multi-party computation are assumed to perform better over high latency networks, the communication overhead of exchanging garbled circuits in these protocols significantly impacts their throughput. In contrast, secret-shared based protocols have lightweight communication cost and can be highly parallelized.

In recent years, there has been increasing research on multi-party computation based on secret-sharing. The main bottleneck in the performance of these protocols is computing arithmetic multiplication and/or Boolean AND gates as it requires the parties to communicate with each other. In the case of 2-party computation, Beaver [3] proposed to divide the computation into a pre-processing phase and an online phase. In the pre-processing phase, the parties generate so called multiplication triples to be consumed in the online phase. However, the pre-processing phase involves heavy cryptographic operations such as oblivious transfer (OT) [40] or homomorphic encryption (HE) [16] to generate the multiplication triples which limits the overall performance of the protocol. It is also not always possible for the parties to be present prior to the execution of the protocol such as in ad-hoc applications.

To avoid such a computationally heavy pre-processing, many protocols have been proposed for 3-party computation. The protocol by Araki et al. [1] leverages a replicated secret-sharing scheme and only requires 1 bit of communication per AND gate per party and uses only simple field operations in the semi-honest security model. The semi-honest protocol version of ASTRA [12] further improves throughput with total 2 bits of communication necessary per AND gate (less than 1 bit per party) in the online phase. But there is no publicly available implementation to replicate their results and compare to. Other protocols exist [13, 15, 20, 33, 43] that extend recent secret-shared based computation to the malicious security model. However, semi-honest majority security has been deployed by many in practice [7, 28, 51] and is sufficient wherever the parties are unlikely to be malicious but want to keep their data private. Moreover, as Goyal and Song [24] show, malicious security comes for free in honest-majority MPC.

In this paper we investigate the reduction of the round complexity of secret-shared based multi-party computation of circuits composed of binary gates in the semi-honest model. We propose a new multi-party computation protocol for three parties (3PC) with an honest majority that a) has a communication cost of 1 bit per party per 2-fan-in AND gates and b) allows to compute multi-fan-in gates in one round, all without any pre-processing. For l-fan-in gates our protocol's communication cost is $2^l - l - 1$ (parties P1 and P2) or 2 (party P3) bits. However, any l-fan-in gate can replace up to $l - 1$ 2-fan-in AND gates. Previous proposals for two-party protocols supporting multi-fan-in gates [41, 42] exists but they require function-dependent pre-processing unlike our approach (no pre-processing). This limitation is inherent to the nature of two-party protocols. Any two-party protocol based on secret shares necessarily requires a pre-processing phase using public-key cryptography, whereas multi-party protocols can be based on information-theoretic assumptions alone.

Proceedings on Privacy Enhancing Technologies 2023(4), 224–237
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2023-0107

To take advantage of the improved performance of our multi-party computation protocol, we design improved circuits that use multi-fan-in AND gates. Previous works such as [11, 47] aim to develop compilers or logic synthesis toolboxes that output Boolean circuits with lower multiplicative depth tailored toward multi-party applications. The multiplicative depth of the circuit is especially important for secret-shared based multi-party computation as each level of the circuit adds a round trip time (RTT) latency to the running time of the protocol. Although, these solutions are automated, we found manual design a better approach to build circuits with multi-fan-in AND gates, since it provides us with more flexibility to choose the optimal fan-in count resulting in the shallowest construction. Other works that have considered multi-fan-in AND gates such as [41, 42] follow the same line. However, in both the fan-in count is limited to 4 due to the limitations of the underlying protocol whereas we allow for upto 8 fan-in AND gates. In particular, we design a new multi-fan-in adder based on Sklansky's design [25] which reduces multiplicative depth of adding two 64-bit numbers from 7 up to 3 (Section 4). We also design a new multi-fan-in multiplier based on Wallace's design [48] which reduces the multiplicative depth of multiplying two 64-bit numbers from 18 up to 8. Finally, we build a circuit for the comparison functionality for which the multi-fan-in gates are a natural choice (Section 4) and reduce the multiplicative depth of the circuit comparing two 64-bit numbers from 7 to 3. The comparison circuit is especially useful in multi-party computation where it finds many applications in sorting [27], PSI [45], auctions [7] and Privacy Preserving Machine Learning (PPML) [38, 39]. We chose to implement all our functions using binary gates, since it avoids share conversions. In hybrid protocols that use both binary and arithmetic sharing, a conversion from arithmetic to binary (and back) is often needed before (and after) comparison, because the comparison is best performed over binary circuits. Conversions between arithmetic and binary shares are costly and in many cases cause loss of precision [39] and limit the number of sequential operations that can be performed. For example, both [38, 39] introduce truncation algorithms that require pre-processing and [12] keep their circuits to only one level of multiplication.

Our protocol remains efficient especially over WAN. For the common benchmark of computing AES cipher blocks, our protocol is the first to achieve sub-second latency over WAN in our setup for all key sizes. We implement our protocol in the MP-SPDZ framework [29] and experimentally compare it to the protocol by Araki et al. [1], Beaver et al.'s technique [4] using replicated secret sharing (as Araki et al. use) [31] and the online phase of ABY2 [42]. The online phase of the two-party protocol ABY2 requires no public-key cryptographic primitives similar to 3PC. Our protocol outperforms these protocols in simulated and real WANs. We use different Amazon data centers connected over the Internet as the real WAN. We improve throughput of AES-128 by a factor of more than 2.2× compared to the best competitor. Our protocol is particularly efficient in case of a real WAN, since it has asymmetric communication allowing a high-latency connection for one party. This leads to much more efficient protocols on these (real) WANs.

In summary, we contribute

- A new honest-majority three-party multi-party computation protocol secure in the semi-honest model that allows to compute multi-fan-in AND gates in one round without precomputation (Section 3)
- New designs of multi-fan-in circuits for addition, multiplication and comparison (Section 4)
- An experimental evaluation of our protocol in the MP-SPDZ framework comparing it to related work [1, 4, 42] (Section 5). We show our protocol outperforms both garbled circuit [4] and secret-shared based [1] protocols with similar setup as ours in running-time and throughput over WAN for the common benchmark of AES. (Section 5)

2 PRELIMINARIES

In this section, we describe our parameters, notation, secret sharing scheme, our correlated randomness setup and the message passing technique of our framework.

2.1 Notation

In our protocol, we have three parties $\mathcal{P} = \{P_1, P_2, P_3\}$ which are connected by standard bidirectional, secure and authenticated channels e.g. via TLS over TCP/IP. Let $P_{i\pm 1}$ refer to the next (+) or previous (-) party with wrap-around, i.e., P_{3+1} is P_1 and P_{1-1} is P_3.

We work over the ring $\mathbb{Z}_{2^n}$. We present protocols for $n = 1$, i.e., bits, but it is possible to extend our protocols to $n > 1$. We use κ to refer to the computational security parameter.

2.2 Correlated Randomness

Our protocol relies on the fact that each pair of parties $\{P_i, P_{i+1}\}$ can obtain a fresh random element r on demand, without any interaction beyond a short initial setup.

To enable this, let $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^n$ be a pseudorandom function (PRF), mapping strings into the ring $\mathbb{Z}_{2^n}$. The function F can be instantiated, for instance, using AES in the counter mode.

Preprocessing:

(1) **Init**: Each party P_i
- Samples $S_{i,i-1}, S_{i,i+1} \in \{0,1\}^\kappa$
- Sends $S_{i,i-1}$ to P_{i-1} and $S_{i,i+1}$ to P_{i+1}.

(2) **Setup**: Each party P_i
- Sets $\mathbf{R}_{i-1}(x) = \mathbf{F}_{S_{i+1,i}}(x) \oplus \mathbf{F}_{S_{i,i+1}}(x)$
- Sets $\mathbf{R}_{i+1}(x) = \mathbf{F}_{S_{i-1,i}}(x) \oplus \mathbf{F}_{S_{i,i-1}}(x)$

Online:

(3) **GenNextRandom**: Parties P_i and P_j
- Party P_i computes $r_{i,j} = \mathbf{R}_{i-1}(id_{i,j})$
- Party P_j computes $r_{i,j} = \mathbf{R}_{j+1}(id_{i,j})$

(without loss of generality $j = i + 1$)

Figure 1: Correlated randomness functionality

Figure 1 shows how the parties interact in the preprocessing phase to share random seeds for the PRFs. During the online phase, parties (P_i, P_j) call **GenNextRandom** to obtain the random value

$r_{i,j} = r_{j,i}$. It is important to note that party P_k is oblivious to the value $r_{i,j}$.

We remark that by using a pseudo-random function, we can only provide security against computationally bounded adversaries and hence our overall protocol is computationally secure.

2.3 Message Masking

We use the correlated randomness from Section 2.2 to mask the messages passed between the parties. Each party P_i runs the functionality **Preprocessing** twice to obtain two distinct pairs of correlated PRFs (R_{i-1}, R_{i+1}) and (R'_{i-1}, R'_{i+1}) which we use to define the masking functions:

$$\begin{aligned} M_{i\to i+1} &= R_{i+1} \\ M_{i\to i-1} &= R'_{i-1} \\ M_{i+1\to i-1} &= R_{i-1} \\ M_{i-1\to i+1} &= R'_{i+1} \end{aligned} \tag{1}$$

Mask: To send a value v from P_i to P_j

(1) P_i invokes $M_{i\to j}$ to get $m_{i\to j}$.
(2) P_i computes $c = v + m_{i\to j}$ and sends it to P_j.

Figure 2: Message passing with correlated randomness

As shown in Figure 2, for each interaction between the pair of parties (P_i, P_j), the sender P_i masks its message v with the value $m_{i\to j}$ before sending it to P_j. We remark that the other party P_k also knows the mask $m_{i\to j}$ due to the properties of the correlated randomness setup and hence the value v is now additively secret-shared between P_j and P_k.

2.4 Secret Sharing

We define a 2-out-of-3 secret sharing scheme, denoted by π_2^3-sharing as follows. In order to share a secret $x \in \mathbb{Z}_{2^n}$, the dealer samples two random elements $\alpha, \beta \in Z_{2^n}$ and distributes the shares such that:

- P_1's share is the pair $(x + \alpha, \beta)$.
- P_2's share is the pair $(x + \beta, \alpha)$.
- P_3's share is the pair (α, β).

Notice that any two shares suffice to recover x. . Table 1 summarizes the individual shares of the servers for a secret x and the necessary PRFs obtained during the preprocessing phase.

	π_2^3-sharing	Rand	Mask
P_1	$(x+\alpha, \beta)$	R_3, R_2	$M_{1\to2}, M_{1\to3}, M_{2\to3}, M_{3\to2}$
P_2	$(x+\beta, \alpha)$	R_1, R_3	$M_{2\to3}, M_{2\to1}, M_{3\to1}, M_{1\to3}$
P_3	(α, β)	R_3, R_1	$M_{3\to1}, M_{3\to2}, M_{1\to2}, M_{2\to1}$

Table 1: Different shares and PRFs held by the parties

We use the following Lemma in the security proof which is straight-forward to see.

Lemma 2.1. *For any two values $x_1, x_2 \in \mathbb{Z}_{2^n}$, and for any $i \in \{1, 2, 3\}$ the distribution over P_i's share of x_1 is identical to the distribution over P_i's share of x_2.*

3 OUR PROTOCOL

In this section, we describe our protocol for three-party computation. Our protocol works for arithmetic circuits over the ring $\mathbb{Z}_{2^n}$ with Boolean circuits being a special case ($n = 1$). The main advantage of our protocol over related work [1, 12, 41, 42] is its ability to compute multi-input multiplication gates (multi-fan-in AND gates in the Boolean case) for 3-party computation in one round of communication without any precomputation.

3.1 Boolean Circuits

In order to simplify the illustration, we start by describing the protocol for the special case of Boolean circuits with AND and XOR gates. We assume three parties $\{P_1, P_2, P_3\}$ that have correlated randomness setup as described in Section 2.2 and use the message passing technique as described in Section 2.3.

3.1.1 ***XOR (addition) gates.*** Let $(x_1+\alpha_1, \beta_1), (x_1+\beta_1, \alpha_1), (\alpha_1, \beta_1)$ be a secret sharing of x_1, and $(x_2+\alpha_2, \beta_2), (x_2+\beta_2, \alpha_2), (\alpha_2, \beta_2)$ be a secret sharing of x_2. In order to compute a secret sharing of $x_1 + x_2$, each party locally computes the addition of its corresponding shares (no communication needed):

- P_1 computes $x_1 + \alpha_1 + x_2 + \alpha_2$ and $\beta_1 + \beta_2$ and outputs $$\Big((x_1 + x_2) + (\alpha_1 + \alpha_2), (\beta_1 + \beta_2)\Big)$$
- P_2 computes $x_1 + \beta_1 + x_2 + \beta_2$ and $\alpha_1 + \alpha_2$ and outputs $$\Big((x_1 + x_2) + (\beta_1 + \beta_2), (\alpha_1 + \alpha_2)\Big)$$
- P_3 computes $\alpha_1 + \alpha_2$ and $\beta_1 + \beta_2$ and outputs $$\Big((\alpha_1 + \alpha_2), (\beta_1 + \beta_2)\Big)$$

It is straight-forward to verify that the above is a valid π_2^3-sharing of $x_1 + x_2$.

3.1.2 ***AND (multiplication) gates.*** We begin by describing the protocol for computing a 2-fan-in AND gate. This requires each party to send a single bit, which seems to be the optimal achievable bandwidth in the information-theoretic, three-party setting [1] without preprocessing. Let x_1 and x_2 be secret-shared among the parties as before.

(1) **Step 1 - Compute (3,3)-sharing**:
  - P_1 computes $v_1 = (x_1 + \alpha_1)(x_2 + \alpha_2)$.
  - P_2 computes $v_2 = (x_1 + \beta_1)\alpha_2 + (x_2 + \beta_2)\alpha_1$.
  - P_3 computes $v_3 = \alpha_1\beta_1 + \alpha_2\beta_2 + \alpha_1\alpha_2$.

We observe that (v_1, v_2, v_3) constitute a $(3, 3)-$sharing of $t = x_1x_2$. However, to proceed further in the protocols, we need to generate a π_2^3-sharing of t.

(2) **Step 2 - Communication**:
  - P_1 sends $c_1 = v_1 + m_{1\to2}$ to P_2.
  - P_2 sends $c_2 = v_2 + m_{2\to1}$ to P_1.
  - P_3 sends $c_3 = v_3 + m_{3\to1}$ to P_1.
(3) **Step 3 - Compute π_2^3-sharing**:
  - P_1 computes $(t + \alpha_t, \beta_t) = \begin{cases} v_1 + c_2 + c_3 \\ c_3 + m_{1\to2} \end{cases}$

- P_2 computes $(t+\beta_t, \alpha_t) = \begin{cases} v_2 + c_1 + m_{3\to 1} \\ m_{2\to 1} + m_{3\to 1} \end{cases}$
- P_3 computes $(\ \alpha_t\ ,\ \beta_t\ ) = \begin{cases} m_{2\to 1} + m_{3\to 1} \\ c_3 + m_{1\to 2} \end{cases}$

In order to show that the result is a valid π_2^3-sharing of x_1x_2, we need to show that both $(t+\alpha_t)+\alpha_t$, and $(t+\beta_t)+\beta_t$ are equal to x_1x_2. This can be demonstrated as follows:

$$\begin{aligned}(t+\alpha_t)+\alpha_t &= v_1 + c_2 + c_3 + \alpha_t \\ &= v_1 + v_2 + v_3 + m_{2\to 1} + m_{3\to 1} + \alpha_t \\ &= x_1x_2 + \alpha_t + \alpha_t \\ &= x_1x_2\end{aligned}$$

Where the first equality is by the definition of the shares, second equality stems from the definitions of c_2 and c_3 and the third equality can be derived from the combination of **Step 1** and the definition of α_t. It can be shown that the second equation $x_1x_2 = (t+\beta_t)+\beta_t$ holds similarly.

3.1.3 *Multi-fan-in AND (multiplication) gates.*

3-Fan-in AND gate. Since the method described in Section 3.1.2 can not be trivially generalized to compute ℓ-fan-in AND gates with $\ell \geq 3$, to simplify the exposition, we start by showing how to compute a 3-fan-in AND gate. Let x_1, x_2, and x_3 be the inputs to a 3-fan-in AND gate where each x_i is π_2^3-shared among the set of parties $\{P_1, P_2, P_3\}$.

To output a valid π_2^3-sharing of the product $t = x_1x_2x_3$, the protocol requires the parties to compute a (2,2)-sharing $(t+\alpha_t, \alpha_t)$ between P_1 and $\{P_2, P_3\}$ following the protocol described in Figure 3.

In order to see the correctness, Observe that

$$\begin{aligned} &v_2^2\beta_3 + v_2^3\beta_2 + v_2^4\beta_1 \\ &\quad = x_1x_2\beta_3 + x_1x_3\beta_2 + x_2x_3\beta_1 + \beta_1\beta_2\beta_3 \end{aligned} \tag{2}$$

Thus,

$$\begin{aligned} &c_2^1 + c_2^2\beta_3 + c_2^3\beta_2 + c_2^4\beta_1 \\ &= \quad v_2^1 + v_2^2\beta_3 + v_2^3\beta_2 + v_2^4\beta_1 \\ &\quad + m_{2\to 1}^1 + m_{2\to 1}^2\beta_3 + m_{2\to 1}^3\beta_2 + m_{2\to 1}^4\beta_1 \\ &= \quad x_1x_2x_3 + x_1\beta_2\beta_3 + \beta_1x_2\beta_3 + \beta_1\beta_2x_3 \\ &\quad + m_{2\to 1}^1 + m_{2\to 1}^2\beta_3 + m_{2\to 1}^3\beta_2 + m_{2\to 1}^4\beta_1 \end{aligned} \tag{3}$$

Where the first equality stems from the definitions of $c_2^1, ..., c_2^4$ by P_2 and the second equality is derived by expanding the terms of $(x_1+\alpha_1)(x_2+\alpha_2)(x_3+\alpha_3)$ and cancelling the repeated values from Eq. (2). Therefore, P_1 computes

- **Compute (2,2)-sharing of $x_1x_2x_3$**
  1. P_2 computes
  $$\mathbf{v}_2 = \begin{cases} v_2^1 = (x_1+\beta_1)(x_2+\beta_2)(x_3+\beta_3) \\ v_2^2 = (x_1+\beta_1)(x_2+\beta_2) \\ v_2^3 = (x_1+\beta_1)(x_3+\beta_3) \\ v_2^4 = (x_2+\beta_2)(x_3+\beta_3) \end{cases}$$
  2. P_2 sends $c_2^i = v_2^i + m_{2\to 1}^i$ to P_1 for all v_2^i.
  3. P_1 computes
  $$\begin{aligned} t+\alpha_t = {} & c_2^1 \\ & + c_2^2\beta_3 + c_2^3\beta_2 + c_2^4\beta_1 \\ & + (x_1+\alpha_1)\beta_2\beta_3 + \beta_1(x_2+\alpha_2)\beta_3 \\ & + \beta_1\beta_2(x_3+\alpha_3) \\ & + m_{3\to 2}^1 \end{aligned}$$
  4. P_3 computes
  $$\begin{aligned} v_3^1 = {} & m_{2\to 1}^1 + m_{2\to 1}^2\beta_3 + m_{2\to 1}^3\beta_2 + m_{2\to 1}^4\beta_1 \\ & + \alpha_1\beta_2\beta_3 + \beta_1\alpha_2\beta_3 + \beta_1\beta_2\alpha_3 \end{aligned}$$
  5. P_3 sends $c_3^1 = v_3^1 + m_{3\to 2}^1$ to P_2 and sets $\alpha_t = c_3^1$.
  6. P_2 sets $\alpha_t = c_3^1$.

Figure 3: Computing (2,2)-sharing of the product $x_1x_2x_3$

$$\begin{aligned} & t+\alpha_t \\ = \quad & x_1x_2x_3 + x_1\beta_2\beta_3 + \beta_1x_2\beta_3 + \beta_1\beta_2x_3 \\ & + m_{2\to 1}^1 + m_{2\to 1}^2\beta_3 + m_{2\to 1}^3\beta_2 + m_{2\to 1}^4\beta_1 \\ & + (x_1+\alpha_1)\beta_2\beta_3 + \beta_1(x_2+\alpha_2)\beta_3 + \beta_1\beta_2(x_3+\alpha_3) \\ & + m_{3\to 2}^1 \\ = \quad & x_1x_2x_3 + m_{2\to 1}^1 + m_{2\to 1}^2\beta_3 + m_{2\to 1}^3\beta_2 + m_{2\to 1}^4\beta_1 \\ & + \alpha_1\beta_2\beta_3 + \beta_1\alpha_2\beta_3 + \beta_1\beta_2\alpha_3 + m_{3\to 2}^1 \\ = \quad & x_1x_2x_3 + c_3^1 \end{aligned}$$

Where the first equality holds by substituting with the equivalence from Eq. (3), the second equality is derived by cancelling the values $x_1\beta_2\beta_3$, $\beta_1x_2\beta_4$, $\beta_1\beta_2x_3$, and the final equality is derived by the definition of c_3^1 by P_3.

However, the multiplication protocol is not complete as the parties must have a π_2^3-sharing of t. To complete the protocol, the parties run a symmetric computation with the roles of P_1 and P_2 exchanged. This will output a (2,2)-sharing $(t+\beta_t, \beta_t)$ between P_2 and $\{P_1, P_3\}$. Together, the two (2,2)-sharings constitute a valid π_2^3-sharing of t.

3.1.4 *ℓ-Fan-in AND (multiplication) gates.*

In this subsection, we describe the protocol to compute ℓ-fan-in AND gates for the general case $\ell \geq 3$. Similar to the 3-fan-in case, our protocol constructs two (2,2)-sharings of the product $t = x_1...x_\ell$ between the

- **Compute (2,2)-sharing** $x_1...x_\ell$

(1) P_2 computes

$$v_{\phi(I)} = \prod_{x_i \in I} (x_i + \beta_i)$$

for $I \subseteq \mathcal{P}(X)$ and $|I| > 1$,

(2) P_2 sends $c_2^{\phi(I)} = v_2^{\phi(I)} + m_{2\to1}^{\phi(I)}$ to P_1 for all $v_2^{\phi(I)}$.

(3) P_1 computes

$$\begin{aligned} t + \alpha_t = &\sum_{I\subseteq X, |I|>1} c_2^{\phi(I)} \prod_{x_j \notin I} \beta_j \\ + &\sum_{x_i \in X, I=\{x_i\}} (x_i + \alpha_i) \prod_{x_j \notin I} \beta_j \\ + &\; b \prod_{i \in \mathbb{L}} \beta_i + m_{3\to2}^1 \end{aligned}$$

where $b = 1$ if n is even o.w. $b = 0$.

(4) P_3 computes

$$\begin{aligned} v_3^1 = &\sum_{I\subseteq X, |I|>1} m_{2\to1}^{\phi(I)} \prod_{x_j \notin I} \beta_j \\ + &\sum_{x_i \in X, I=\{x_i\}} \alpha_i \prod_{x_j \notin I} \beta_j \end{aligned}$$

(5) P_3 sends $c_3^1 = v_3^1 + m_{3\to2}^1$ to P_2 and sets $\alpha_t = c_3^1$.

(6) P_2 sets $\alpha_t = c_3^1$.

Figure 4: Computing (2,2)-sharing of the product $x_1...x_\ell$

parties. In the following, we only present the (2,2)-sharing where P_1 holds $t + \alpha_t$ and $\{P_2, P_3\}$ both hold (α_t). Since the roles of P_1 and P_2 are symmetric, the full protocol follows naturally.

Let $\mathbb{L} = \{1, ..., \ell\}$, $2^{\mathcal{L}} = \{0, ..., 2^\ell - 1\}$, and $\mathcal{P}(X)$ be the power set of the set of inputs $X = \{x_1, ..., x_\ell\}$. Define $\phi : \mathcal{P}(X) \to 2^{\mathcal{L}}$ as $\phi(I) = \Sigma_{x_i \in I} 2^{i-1}$. Parties $\{P_1, P_2, P_3\}$ hold a π_2^3-sharing of each x_i such that P_1 holds $(x_i + \alpha_i, \beta_i)$, P_2 holds $(x_i + \beta_i, \alpha_i)$ and P_3 holds (α_i, β_i). The protocol for computing a (2,2)-sharing of $t = x_1...x_\ell$ is presented in Fig. 4.

To show correctness, we need the following lemma.

Lemma 3.1. *Assume* $\{x_1, ..., x_\ell\}$ *and* $\{\beta_1, ..., \beta_\ell\}$ *are two sets of values over* $\mathbb{Z}_2$. *Then*

$$\begin{aligned} &\sum_{I\subseteq X, |I|>1} \prod_{x_i \in I} (x_i + \beta_i) \prod_{x_j \notin I} \beta_j \\ &= \prod_{i \in \ell} x_i + \sum_{x_i \in X, I=\{x_i\}} x_i \prod_{x_j \notin I} \beta_j + b \prod_{i \in \ell} \beta_i \end{aligned}$$

where $b = 1$ *if* ℓ *is even,* $b = 0$ *otherwise.*

Proof. Let $X_I = \prod_{x_i \in I} x_i \prod_{x_j \notin I} \beta_j$ for every I where $1 < |I| < \ell$. It is straight-forward to see that X_I is a factor of the term

$$\sum_{I\subseteq A \subseteq \mathcal{P}(X)} \prod_{x_i \in A} (x_i + \beta_i) \prod_{x_j \notin A} (\beta_j)$$

The number of sets $A \in \mathcal{P}(X)$ where $I \subseteq A$ is $2^{\ell - |I|}$. Since $1 < |I| < \ell$, it follows that each X_I is repeated an even number of times on the left side of the equation, hence, it gets cancelled. Following the same argument, observe that terms of the form $x_i \prod_{x_j \neq x_i} \beta_j$ occur a total $2^{\ell-1} - 1$ times where the subtraction is due to $\{x_i\}$ being excluded from the sum. Similarly, $x_1...x_\ell$ only appears once and $\prod_{i\in\mathbb{L}} \beta_i$ occurs $2^\ell - \ell - 1$ times. Thus, completing the proof. ■ □

Therefore, the correctness of the protocol follows from

$$\begin{aligned} t + \alpha_t = &\sum_{I\subseteq X, |I|>1} \prod_{x_i \in I} (x_i + \beta_i) \prod_{x_j \notin I} \beta_j \\ + &\sum_{I\subseteq X, |I|>1} m_{2\to1}^{\phi(I)} \prod_{x_j \notin I} \beta_j \\ + &\sum_{x_i \in X, I=\{x_i\}} x_i \prod_{x_j \notin I} \beta_j + \sum_{x_i \in X, I=\{x_i\}} \alpha_i \prod_{x_j \notin I} \beta_j \\ + &\; b \prod_{i \in \mathbb{L}} \beta_i + m_{3\to2}^1 \qquad (2) \\ = &\prod_{i \in \mathbb{L}} x_i + \sum_{I\subseteq X, |I|>1} m_{2\to1}^{\phi(I)} \prod_{x_j \notin I} \beta_j \\ + &\sum_{x_i \in X, I=\{x_i\}} \alpha_i \prod_{x_j \notin I} \beta_j + m_{3\to2}^1 \\ = &\prod_{x_i \in \mathbb{L}} x_i + c_3^1 \end{aligned}$$

where the first equality is derived by the definition of $c_2^{\phi(I)}$, the second equality is derived by substituting Lemma 3.1, and the final equality is derived from the definition of c_3^1.

3.2 Threat Model and Security

In this subsection, we prove the security of our protocol in the presence of a static semi-honest adversary controlling one of the three parties. Since our goal is to compute a Boolean circuit, the functionality is deterministic and we can use the following security definition by Goldreich [21]:

Definition 3.2. Let $f : (\{0,1\}^*)^3 \to (\{0,1\}^*)^3$ *be a deterministic 3-ary functionality and let* π *be a protocol. We say that* π *securely computes* f *in the presence of one static semi-honest adversary if for every* $\vec{x} \in (\{0,1\}^*)^3$ *where* $|x_1| = |x_2| = |x_3|$ *the following holds:*

1) Correctness: $OUTPUT^\pi(\vec{x}) = f(\vec{x})$, *and*

2) Privacy: *There exists probabilistic polynomial time algorithms* $\mathcal{S}_i$ *such that for every corrupted party* $i \in \{1,2,3\}$:

$$\mathcal{S}_i(x_i, f_i(\vec{x})) = \{VIEW_i^\pi(\vec{x})\}$$

Where $VIEW_i^\pi(\vec{x})$ *is the view of party i and consists of its input* x_i, *its internal random coins* r_i *and the messages received by i during the protocol execution.*

Proof. The correctness requirement of our protocol follows immediately as the parties compute the circuit C which implements functionality f and we have already shown correctness for the AND (multiplication) and XOR (addition) gates composing the circuit.

It remains to show privacy. If party P_3 is corrupted, note that it receives nothing during the execution of the protocol both for

XOR gates and AND gates. The only messages that P_3 receives are the setup bits for the correlated randomness from P_1 and P_2 which are just two independent random bits $S_{1,2}, S_{2,1} \in \{0,1\}^{\kappa}$. Thus the simulator $\mathcal{S}_3$ simply chooses two random bits $S'_{1,2}, S'_{2,1}$, sets up the pseudo-random functions with them. For each output wire in which P_3 receives output, given the shares computed thus far by the simulator and the true output value, $\mathcal{S}_3$ can generate the exact shares it would receive from the other parties. This is because of the secret sharing scheme's property that given one party's shares and the exact value of the secret, the associated parties' shares can be fully determined. Thus, the simulator's view is completely indistinguishable form a real execution of π.

If party P_1 is corrupted, unlike P_3 it receives messages during the protocol execution from both parties P_2 and P_3. The view of P_1 during the protocol execution consists of the random bits $S_{2,3}, S_{3,2}$ it receives from P_2 and P_3 respectively for the correlated randomness setup and the messages it receives from them for 2-fan-in and ℓ-fan-in AND gates. The simulator $\mathcal{S}_1$ simulates the correlated randomness setup bits as in the case of P_3. For any AND gate g in the circuit, P_1 receives messages depending on the gate's fan-in:

(1) If g is a 2-fan-in AND gate, then P_1 receives one single bit from P_2 which is $(x_1 + \beta_1)\alpha_2 + (x_2 + \beta_2)\alpha_1 + m_{2\to1}$ where x_1 and x_2 are the the actual inputs to gate l and $m_{2\to1}$ is the randomness P_2 uses for communication with P_1 on that gate. Since the bit is masked with the random bit $m_{2\to1}$ obtained from the shared pseudo-random function between P_2 and P_3 that is oblivious to P_1, the simulator $\mathcal{S}_1$ simply samples a random bit and simulates the above message.
(2) If g is a ℓ-fan-in AND gate, then P_1 receives $2^\ell - \ell - 1$ bits from P_2 and one bit from P_3 as described in the Section 3.1.4. However, all these messages are masked using random bits as in the 2-fan-in case unknown to P_1. Therefore, $\mathcal{S}_1$ simulates such gates by sampling $2^\ell - \ell$ random bits.

Finally, for each output wire $\mathcal{S}_1$ does the same thing as $\mathcal{S}_3$.

If P_2 is corrupted, we observe that the views of P_1 and P_2 on ℓ-fan-in AND gates and the correlated randomness setup bit is symmetric. The only difference in the view of these two parties occurs when computing a 2-fan-in AND gate where P_2 receives an additional bit from P_3 that P_1 does not. However, since this bit is also masked with a random bit unknown to P_2, the simulator for P_2 can simply sample an additional random bit.

Thus, we can construct a simulator for all static adversaries, hence, from this and by the security of the pseudo-random function, the simulator-generated views are identically distributed to that of a real protocol execution. ■

4 COMMUNICATION-EFFICIENT CIRCUITS

In this section, we describe how to use multi-fan-in AND gates to design binary circuits with lower multiplicative depth.

Specifically, we focus on adder, multiplier and comparator circuits.

In the following, the depth of the circuit denotes the number of multiplicative stages and the size is the total number of multiplications.

Adder: An n-bit adder takes two n-bit numbers a and b and outputs a n-bit number s plus a carry out c. The simplest construction for an adder is the Ripple-Carry adder which has minimal size but due to its sequential structure has a multiplicative depth of n. Therefore, it is not well-suited for MPC purposes.

Parallel Prefix Adders (PPA) [25] use prefix tree networks with generate and propagate signals to obtain lower depth circuits with a trade-off against circuit size. There are many PPA designs with different characteristics. However, Sklansky PPA achieves the lowest depth without excessively increasing circuit size. Using traditional Sklansky PPA with 2-fan-in gates results in a circuit depth of $\log_2 n$. Due to the associativity of propagate and generate signals, taking advantage of higher-fan-in gates is straightforward. This yields a higher parallelization and decreases the depth of the circuit by a factor of $\log_2 \ell$.

As our protocol has communication complexity exponential in the AND gate fan-in, we limit ourselves to Sklansky structures that are based on $\ell \leq 8$-fan-in AND gates. Table 2 compares the depth and size of the circuits for different bit-widths for the Ripple-Carry adder, an unmodified Sklansky PPA adder and our two new constructions with multi-fan-in AND gates up to $\ell \leq 4$ and $\ell \leq 8$, respectively.

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Ripple-Carry	16	32	64	128	31	63	127	255
Sklansky	5	6	7	8	65	161	385	897
$\ell \leq$4-fan-in	3	4	5	8	73	177	433	993
$\ell \leq$8-fan-in	3	3	3	4	87	213	561	1249

Table 2: Comparison of adder circuit depth and size for different constructions and bit-widths

Multiplier: An n-bit multiplier takes two n-bit numbers a and b and outputs a $2n$-bit number p as their product. The common algorithm for multiplication is the same as the high-school method where first n partial products of bit-length n are computed by multiplying each bit of the multiplier by the multiplicand and left-shifting the result by the position of the multiplier's bit. The basic multiplier circuit then uses a $2n$-bit adder to sum the partial products one after one until the final product is reached.

Even with our depth-optimized adders, this approach has a $n \log_\ell n$ depth when sequentially adding the partial products and using 2-fan-in AND gates in the adder design. Summing up the partial products in a binary tree reduces the number of addition levels from n to $\log n$ but it is still expensive in MPC protocols.

In logic synthesis, a Wallace tree [48] multiplier is used to gather all the partial products in a tree structure. Then, each column of bits are divided in groups of 3 and reduced to 2 bits using Full Adders and Half Adders. This process is repeated until there are only 2 bits left at each position whereby a final $2n$-bit adder is used to output the product. A Wallace tree multiplier therefore has a depth of $\log_{1.5} n + \log_\ell 2n + 1$ where the former is the number of reduction stages and the latter is due to the final adder.

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Standard	45	93	189	381	496	2016	8128	32640
Wallace	13	15	18	21	512	2058	8226	32836
$\ell \leq$4-fan-in	7	9	9	11	1229	5304	21997	89416
$\ell \leq$8-fan-in	6	6	8	11	1422	6612	24834	96679

Table 3: Comparison of multiplier circuit depth and size for different constructions and bit-widths

Algorithm	depth				size			
	16	32	64	128	16	32	64	128
Standard	5	6	7	8	63	143	319	703
$\ell \leq$4-fan-in	3	4	4	5	39	95	207	479
$\ell \leq$8-fan-in	3	3	3	4	37	83	175	415

Table 4: Comparison of comparator circuit depth and size for different constructions and bit-widths

There are many optimization techniques to further reduce the number of stages. Asif and Kong [2] introduce the notion of Counter-Based Wallace (CBW) tree reduction where 4:3, 5:3, 6:3 and 7:3 counters are applied to each column to compress more bits at each stage reducing the number of stages to $\log_{2.3} n$. By a similar extension, using 15:4 counters results in $\log_{3.75} n$ stages and larger counters $N : M$ in $\log_{N/M} n$. However, the counter circuit is not a depth-1 circuit anymore, and, hence, requires additional communication rounds per stage.

We use multi-fan-in AND gates to design depth-1 counter circuits and take full advantage of the fewer reduction stages without incurring extra communication rounds per stage. However, this comes with a trade-off against the size of the circuit and also needs higher fan-in AND gates. Table 3 compares the depth and size of the circuits for different bit-widths for the standard multiplier, the Wallace-tree multiplier and our two new constructions with 7:3 counters and $\ell \leq$ 4-fan-in AND gates and 15:4 counters and $\ell \leq$ 8-fan-in AND gates.

Comparator: A binary circuit to compare two n-bit numbers a, b in secure computation is presented by Fischlin [19]. A number a is greater than another number b if, for some i, the i-th bit of a is 1 and 0 for b and all the more significant bits $> i$ are equal. Formally:

$$a > b \iff \bigoplus_{i=1}^{n} \left(a_i \wedge \neg b_i \wedge \bigwedge_{j=i+1}^{n} (a_j = b_j)\right)$$

where $a_j = b_j$ can be rewritten as $\neg(a_j \oplus b_j)$. Using a tree structure of 2-fan-in AND gates will result in a circuit of depth $\log_2 n + 1$ to compute the product of terms in Fischlin's equation. Similar to the adder circuit, higher fan-in gates can be used to reduce the number of communication rounds by minimizing the depth of the multiplicative tree. Table 4 compares the depth and size of the circuits for different bit-widths for the unmodified comparator and our two new constructions with multi-fan-in AND gates up to $\ell \leq 4$ and $\ell \leq 8$, respectively.

5 EXPERIMENTAL IMPLEMENTATION AND BENCHMARKS

In this section, we provide empirical results of a prototypical implementation demonstrating the efficiency gains due to (our protocol design and) the use of multi-fan-in AND gates compared to Araki et al.'s [1] and Beaver et al.'s protocol [4]. Recall that ASTRA [12], which (slightly) improves over Araki et al.'s protocols, has no public implementation to replicate their results. We begin by describing the setup environment and implementation details.

We implement our protocol using the MP-SPDZ [29] framework in C++11. For our servers, we use three AWS t2.large instances equipped with two Intel Xeon E5-2686 v4 2.3GHz CPUs and 8 GB RAM. We run the experiments in a simulated WAN and a real WAN setting. To simulate the WAN, we run all three parties on a single instance imposing a round-trip time (RTT) of *100ms* and a channel bandwidth of 160Mbps by manipulating the Kernel's Traffic Control for the local interface. Over the WAN, the machines are instantiated in Amazon Web Services data centers at US East (*P1*), US West (*P2*) and North Europe (*P3*). In this setting, the network statistics for each pair *P1-P2*, *P1-P3*, *P2-P3* are measured separately, where the average RTTs are approximately *50ms*, *100ms* and *150ms* with channel bandwidth of approximately 235Mbps, 115Mbps and 75Mbps, respectively. For our AES benchmark we also use a local-area network (LAN) with all 3 parties running on the same machine connected by the loopback interface which has roughly 20 Gbps bandwidth and a RTT of 0.05 ms. We use TLS over TCP for secure communication between each pair of parties.

We compare the performance of our protocol against the closest competitor [1] which shares the same threat model, requires no pre-computation as ours and has a publicly available implementation. We ensure fair comparison by running the MP-SPDZ implementation of Araki et al.'s protocol [1] in our environment. As Araki et al.'s protocol [1] does not support multi-fan-in AND gates, the results are only reported for circuits based on standard 2-fan-in AND gates. We then extend the measurements to multi-fan-in gates for our protocol. We also compare to the implementation of Beaver et al.'s protocol [4] in the MP-SPDZ framework [29] based on the techniques by Keller and Yanai [31]. For Beaver et al.'s technique 2-fan-in gates are optimal, since they have the lowest communication cost and the number of communication rounds cannot be further reduced by increasing the fan-in. Hence, we use circuits with 2-fan-in gates for Beaver et al.'s technique. We use the online running time and communication cost as the measured quantities to benchmark the protocols. To generate both the standard 2-fan-in and the round-optimized circuits using the multi-fan-in AND gates, we implemented a Python module to compile the circuit designs into Bristol Fashion format suitable as input to the MP-SPDZ framework.

We perform the comparison for three types of circuits. First, we begin by analyzing the advantages of multi-fan-in gates in the theoretically optimal case of an AND-tree (Section 5.1). Second, we compare our performance to related work [1, 4] for our depth-optimized building block circuits for addition, multiplication and comparison (Section 5.2). Third, we compare our performance to related work for a practical circuit, the AES block cipher, which has been used in many related works as a benchmark (Section 5.3). For this comparison, we also include the 2-party protocol ABY2 [42], since it also supports multi-fan-in gates. However, we only compare to the online phase (ignoring its offline phase) of ABY2, in order to avoid having to resort to public-key operations. Since, we operate in a 3-party setup our protocol can be and is implemented without public-key cryptography.

5.1 Results for AND Trees

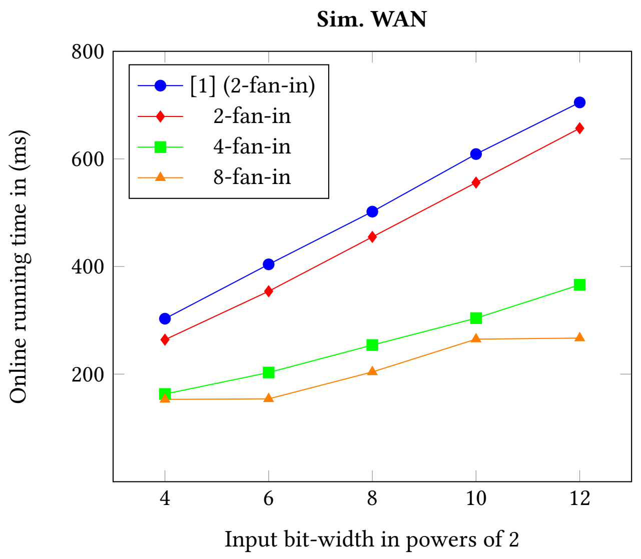
To analyze the performance of 2-fan-in AND gates compared to the multi-fan-in AND gates, we construct a prototype circuit that computes $\bigwedge_{i=1}^{n} a_i$ for an n-bit number a. Similar to the comparison circuits from Section 4, a tree structure of AND gates is used to aggregate the input wires in multiple rounds until the final product is output. The depth and size of this circuit are $\lceil \log_\ell n \rceil$ and $\frac{\ell^{\lceil \log_\ell n \rceil}-1}{\ell-1}$, respectively allowing for ℓ-fan-in AND gates.

Figure 5 shows the running time of the AND-tree aggregator for designs based on 2,4,8-fan-in AND gates and Araki et al.'s protocol [1] using 2-fan-in gates. In the simulated WAN setting, we observe in the case of 2-fan-in gates, that for both Araki et al.'s [1] and our protocol, the running time increases linearly with the multiplicative depth of the circuit. However, our protocol consistently takes one RTT time less to complete which is due to the asymmetry of the roles of the parties. This behavior has been noticed before in other protocols, notably ASTRA [12], where similarly, the majority of the communication and computation is handled by only a pair of the parties, namely *P1-P2*. In our protocol, *P3* does not receive any bits from the other parties, i.e., it can execute the protocol without waiting on any inbound communication. Consequently, by the time *P1* and *P2* reach the final level of the circuit, they have already received the information required from *P3* to do the local computation. Hence, they can finalize the protocol with no latency saving an extra RTT in the running time.

In the real WAN setting, this improvement becomes even more significant as our protocol provides the flexibility to setup the parties such that the communication link between *P1-P2* has the lowest latency among all the pairs. This effectively reduces the delay for each round from the maximum to the minimum RTT among each pair of the parties due to the role of *P3* as explained. In Figure 5, we observe that our protocol achieves better performance by improving up to 1.7× upon Araki et al.'s protocol [1].

Next, we analyze the impact of multi-fan-in AND gates and demonstrate the full power of our protocol. Even when using only a slightly higher fan-in $\ell \leq 4$-AND gates, we achieve an improvement in the running time of 2× for the simulated WAN and 3.5× for the real WAN compared to Araki et al.'s protocol [1]. This is mainly due to the reduced depth of the circuits, though the asymmetry of our protocol contributes to the increased performance over WAN as well.

The running time can be further reduced by using $\ell \leq 8$-fan-in AND gates. The performance gain becomes more noticeable as the circuit sizes grow larger. However, for some input lengths the running time is not improved upon the case of $\ell \leq 4$-fan-in. This is because the circuit depth remains the same regardless of the design using either ≤ 4 or ≤ 8 fan-in AND gates. However, since



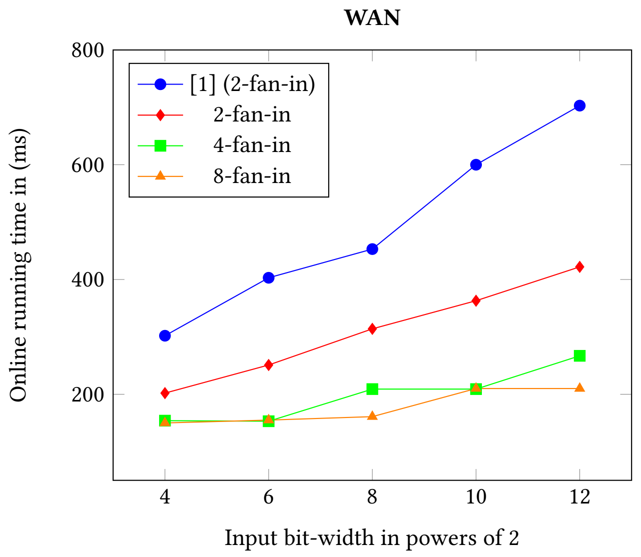


Figure 5: Online latency of AND-tree computation for multiple input bit-width

our protocol has exponential communication cost in the fan-in, the optimal constructions are the ones leveraging smaller fan-in, i.e., $\ell \leq 4$-fan-in gates.

Table 5 shows the communication cost of our protocol for running 100 instances of the AND-tree aggregator in parallel. We report the numbers for all the parties, however, as *P1* and *P2* have symmetric behaviour, they incur the same amount of communication and are thus grouped together. The results for Araki et al.'s protocol [1] are the same as ours in the case of circuits with only 2-fan-in gates as they both send one bit per party per 2-fan-in AND gate. We see that for each input size, the communication cost increases for *P1*, *P2* as higher fan-in gates are used. This is an expected behaviour, since the multi-fan-in AND gates consume exponentially more bits

Party	Fan-in	Comm. (KB)				
		Tree-2^4	Tree-2^6	Tree-2^8	Tree-2^{10}	Tree-2^{12}
P1, *P2*	2	0.013	0.018	0.042	0.138	0.522
	4	0.017	0.039	0.127	0.479	1.887
	8	0.072	0.288	1.123	4.518	18.072
P3	2	0.011	0.017	0.041	0.137	0.521
	4	0.011	0.015	0.031	0.095	0.351
	8	0.010	0.012	0.019	0.046	0.156

Table 5: Comparison of communication cost of AND-tree computation per party for multiple input bit-width

on the link *P1-P2*. This explains why it is not always optimal to use higher fan-in gates, i.e., when they do not sufficiently reduce the multiplicative depth. For example, the 2^{10}-tree aggregator circuit based on 8-fan-in AND gates has 10× more communication cost than the one based on 4-fan-in AND gates, while there is no improvement in running time.

However, we observe that *P3* uses less communication with higher fan-ins. This is because *P3* always sends 2 bits per multi-fan-in AND gates regardless of the fan-in. Therefore, its communication cost gets lower as the circuit sizes shrink due to replacing lower fan-in with higher fan-in gates.

Another observation here is that our protocol has the advantage of being resilient to a lower bandwidth or even a change in bandwidth between one party and the others due to its asymmetry. If one of the parties suffers from a limited bandwidth to the other parties, we can dynamically assign it the role of P3, i.e., the party with the lowest communication overhead. As a consequence, our protocol may maintain its performance in case of bandwidth fluctuations, particularly if high fan-in AND gates are used. This improves over Araki et al.'s protocol [1] which would suffer a significant performance downgrade under such circumstances. Chaudhari et al. have already noted this for ASTRA [12] for two-fan-in gates.

5.2 Results for Communication-Efficient Circuits

We benchmark the latency and communication cost of our depth-optimized circuits for addition, multiplication and comparison from Section 4 in our protocol and related work [1, 4]. We take a multi-step approach. In the first step, we compare common circuits against our depth-optimized circuits using only 2-fan-in gates in our and Araki et al.'s protocol (see Table 6). This demonstrates the advantages of our circuits over non-optimized ones on WAN for either our or Araki et al.'s protocol. Then, in the second step we further optimize these circuits with multi-fan gates (see Table 7). This optimization is only possible in our protocol and demonstrates the advantages of our protocol over Araki et al.'s protocol. The combination of the two approaches – depth-optimized circuits and multi-fan-in gates – yield the best results in our protocol and cannot be matched by Araki et al.'s protocol. In the third step, we compare this combination in our protocol to the common circuits in Beaver et al.'s protocol (see Table 8). Beaver et al.'s protocol performs independent of circuit-depth and best on two-fan-in gates. Hence, we are comparing the best modes for each protocol and the results demonstrate the advantages of our protocol. Next to latency, we also report communication costs (see Tables 9 and 10).

Circuit	Work	n = 16		n = 32		n = 64	
		Sim.	WAN	Sim.	WAN	Sim.	WAN
RC Adder	[1]	0.906	0.903	1.709	1.678	3.322	3.316
	This	0.865	0.507	1.671	0.927	3.281	1.708
Sklansky	[1]	0.352	0.302	0.405	0.401	0.452	0.448
	This	0.312	0.196	0.361	0.255	0.412	0.257
Standard Multiplier	[1]	-	-	6.494	6.492	9.607	9.608
	This	-	-	6.452	3.243	9.571	4.789
Wallace	[1]	0.906	0.911	1.111	1.059	1.317	1.298
	This	0.865	0.512	1.067	0.609	1.272	0.718
Comparator	[1]	0.352	0.351	0.403	0.404	0.453	0.455
	This	0.312	0.203	0.362	0.251	0.413	0.248

Table 6: Comparison of online running time (sec) of our protocol, Araki et al.'s protocol [1] for addition, multiplication and comparison circuits

We start by analyzing the effect of our protocol when using only 2-fan-in AND gates. Although, Büscher and Katzenbeisser [11] also build depth-optimized circuits, they only theoretically analyze them and do not report any experimental results. Table 6 compares the running time of our protocol against Araki et al.'s protocol [1] for both vanilla and our optimized versions of adder, multiplier and comparator circuits. In the simulated WAN setting, the running time is consistently improved but very similar. Over the real WAN, we take advantage of the asymmetry in the communication structure of our protocol to further reduce the total latency as explained. This increases the running time gains of our protocol to up to 2× over Araki et al.'s protocol.

In Table 7, we present the running time of our protocol for the depth-optimized circuits based on multi-fan-in AND gates. The results show a pattern consistent with our observations for the AND-tree aggregators from Section 5.1, e.g., the running time improves as the multiplicative depth of the circuit decreases due to the use of higher fan-in AND gates.

Next, we compare our protocol against constant-round, garbled circuit-based protocols. We compare to the semi-honest 3-party version of the Beaver et al.'s protocol [4], which has comparable security assumption to our protocol. The inputs are divided into replicated secret-shares as in Araki et al.'s protocol [1]. Beaver et al.'s protocol [4] also has a precomputation phase which we do not report. During the precomputation phase, each party P_i garbles the circuit locally. Then, in the online phase, the parties jointly generate a distributed garbled circuit which they subsequently evaluate to obtain their output.

Table 8 summarizes the running time of our protocol and the online phase of Beaver et al.'s protocol [4] for addition, multiplication and comparison. We report only the running times for the best possible circuits for each protocol. In the simulated WAN setting, Beaver et al.'s protocol [4] has a lower running time for a single instance of all functions. However, over the real WAN, our protocol outperforms it in all except the 32-bit addition functionality where it, however, matches it. We observe that, since the best circuits for our protocol are based on multi-fan-in AND gates, the reduced number of communication rounds significantly affects its running time over WAN. Although Beaver et al.'s protocol [4] has a constant number of rounds, the communication cost of distributing the garbled circuit dominates the network latency. Table 9 shows these communication costs for each function in both protocols. Our protocol has a lower communication cost of 30-420× for 32-bit and 18-120× for 64-bit circuits. The difference in communication costs is further underpinned by the running time of batches of 100 parallel instances of the same circuit. In these cases, our protocol is faster in both simulated and real WAN settings and processes a higher number of parallel circuits.

Circuit	Fan	Sim. WAN (s)				WAN (s)			
	In	16	32	64	128	16	32	64	128
Sklansky	2	0.312	0.362	0.412	0.463	0.196	0.255	0.257	0.310
	4	0.211	0.261	0.261	0.312	0.150	0.194	0.201	0.206
	8	0.203	0.212	0.213	0.266	0.150	0.152	0.204	0.203
Wallace	2	0.865	1.067	1.272	1.388	0.512	0.609	0.718	0.865
	4	0.665	0.767	0.925	1.064	0.411	0.461	0.628	0.641
Comp.	2	0.312	0.362	0.413	0.463	0.203	0.251	0.248	0.316
	4	0.211	0.261	0.261	0.313	0.146	0.203	0.202	0.216
	8	0.211	0.211	0.212	0.263	0.151	0.154	0.155	0.217

Table 7: Comparison of online running time (sec) of addition, multiplication and comparison circuits for multiple fan-ins and input bit-width

Finally, in Table 10, we show the communication cost of our protocol for running 1000 instances of each circuit in parallel alongside a breakdown of their numbers of AND gates by fan-in. The circuits are mostly composed of 2-fan-in and $\ell \leq 4$-fan-in AND gates. Hence, despite the exponential communication complexity of multi-fan-in gates in our protocol, the majority of the computation is performed using AND gates of low fan-in which results in an overall sub-exponential increase in communication cost when using our protocol with multi-fan-in gates.

5.3 Results for AES with Optimized S-box

Since the work by Pinkas et al. [44], the running time to evaluate an AES block cipher has been the common benchmark to measure the performance of MPC protocols. In privacy-preserving AES, one party holds the key k, another party holds a message m and the goal is to learn $AES_k(m)$ without revealing any more information about k or m. This has many interesting applications in practice including encrypted databases [9, 34] and secure user authentication [1, 30].

Func.	Work	Batch = 1				Batch = 100			
		Sim.		WAN		Sim.		WAN	
		32	64	32	64	32	64	32	64
Add.	[4]	0.150	0.150	0.150	0.207	0.583	0.652	1.211	1.37
	This	0.212	0.213	0.152	0.204	0.252	0.252	0.252	0.253
Mult.	[4]	0.586	0.639	1.179	1.316	7.493	11.375	20.35	24.64
	This	0.767	0.925	0.461	0.628	0.960	1.422	0.727	1.174
Comp.	[4]	0.150	0.199	0.200	0.351	0.651	0.874	1.448	3.279
	This	0.211	0.212	0.154	0.155	0.251	0.252	0.251	0.249

Table 8: Comparison of online running time (sec) of our protocol and Beaver et al.'s protocol [4] for addition, multiplication and comparison circuits

Function	Work	n = 32	n = 64
Addition	[4]	38.900	78.300
	This	1.325	4.378
Multiplication	[4]	3600.000	4940.000
	This	8.533	37.173
Comparison	[4]	87.050	194.100
	This	0.584	2.010

Table 9: Comparison of communication cost (KB) of our and Beaver et al.'s protocol [4] for addition, multiplication and comparison circuits

Circuit	n = 16				n = 32				n = 64			
	# AND			Comm. (MB)	# AND			Comm. (MB)	# AND			Comm. (MB)
	2	4	8		2	4	8		2	4	8	
Sklanksy	65	-	-	0.056	161	-	-	0.094	385	-	-	0.182
	43	30	-	0.104	107	70	-	0.208	235	198	-	0.504
	39	24	24	0.595	87	74	52	1.325	183	190	188	4.378
Wallace	828	-	-	0.342	3330	-	-	1.282	13098	-	-	4.949
	718	511	-	1.783	2755	2549	-	8.533	10830	11167	-	37.173
Comp.	63	-	-	0.056	143	-	-	0.089	319	-	-	0.161
	8	31	-	0.088	32	63	-	0.154	48	159	-	0.342
	10	19	8	0.291	12	55	16	0.584	16	95	64	2.010

Table 10: Comparison of communication cost of 1000 instances of addition, multiplication and comparison circuits for multiple fan-ins and input bit-width. The communication cost of Araki et al.'s protocol [1] is the same as our protocol for only 2-fan-in gates.

The AES block cipher is implemented in multiple sequential rounds. Each round performs four operations on the input bytes, namely, SubBytes (S-box), ShiftRows, MixColumn and AddRoundKey [14]. Huang et al. show that only the S-boxes require non-linear operations and hence are relevant to the running time of MPC protocols [26]. Boyar et al. present an optimized circuit of depth 4 and size 34 with only standard 2-fan-in AND gates [8]. Utilizing the power of 3-fan-in gates, Patra et al. reduce the depth of this circuit to 3 while keeping the same size [42]. We additionally take advantage of 4-fan-in gates to build a circuit of depth 2. This comes at the cost of increasing the size of the circuit to 66 of which 22 are 2-fan-in, 22 are 3-fan-in and 22 are 4-fan-in AND gates.

Param.	Work	AES-128		AES-192		AES-256	
		Sim.	WAN	Sim.	WAN	Sim.	WAN
Runtime (sec)	[42]	2.006	2.052	2.408	2.455	2.810	2.856
	[1]	1.964	2.071	2.417	2.520	2.720	2.885
	[4]	0.651	1.147	0.673	1.120	0.677	1.124
	This	1.056	0.655	1.258	0.747	1.458	0.858
Comm (KB)	[42]	2.64		3.16		3.69	
	[1]	2.04		2.42		2.85	
	[4]	11680		13080		16110	
	This	165.7		198.2		230.8	

Table 11: Comparison of online running time and communication cost of our protocol and [1, 4] for AES. Results of our protocol and [1, 42] are reported for 100 parallel instances

Table 11 compares the latency and communication cost of our protocol for this AES circuit to those of related work [1, 4, 42]. ABY2 [42] is a two-party protocol, but we only compare to the online phase, which does not require public-key cryptographic operations but only operations on secret shares as our protocol. We use our new depth-optimized circuit for our protocol and ABY2 [42], since they both support multi-fan-in gates. We use the circuit by Boyar et al. [8] (only 2-fan-in gates) for Araki et al.'s [1] and Beaver et al.'s protocol [4]. Our protocol has a higher communication cost than related work [1, 42] as it requires an exponential number of bits in the fan-in. However, in the simulated WAN setting, we outperform the running time of both, ABY2 [42] and Araki et al.'s protocol [1] by 1.9× for AES-128. Our running time advantage is even stronger over WAN where our protocol improves by 1.8× even over Beaver et al.'s protocol [4] due to the effects of its asymmetric communication. We achieve sub-second running time for all the 3 key sizes of AES over the real WAN (Amazon data centers) which is a first for MPC protocols with the same setting as ours. This is particularly important in case of streaming encryption, such as AES-CBC mode.

The other common metric, with which to analyze MPC protocols running AES, is throughput, i.e., the number of AES blocks per second a protocol can compute. Table 12 compares the amortized throughput of our protocol to related work [1, 4, 42] in both LAN and WAN. We use circuits with depth-3 S-boxes built only with

Circuit	Work	LAN	WAN
AES-128	[4]	131.9	3.2
	[42]	75 K	4.8 K
	[1]	115.5 K	8.2 K
	This	111.2 K	18.1 K
AES-192	[4]	119.3	2.4
	[42]	62.5 K	4.2 K
	[1]	95.5 K	6.8 K
	This	91.6 K	14.8 K
AES-256	[4]	89.1	2
	[42]	54.2 K	3.6 K
	[1]	81.3 K	5.8 K
	This	78.4 K	12.2 K

Table 12: Comparison of online throughput of our protocol and [1, 4] for AES.

3-fan-in gates (no 4 fan-in gates) for our protocol and ABY2 [42]. Despite that our new depth-2 S-boxes are shallower, their communication overhead uses too much bandwidth lowering total (amortized) throughput.

Araki et al.'s protocol [1] has the highest throughput in LAN. Our protocol's throughput is slightly lower, but still competitive since we require extra branching instructions and PRF invocations to compute multi-fan-in AND gates. Over WAN, this additional computation becomes less relevant compared to the network latency such that our protocol improves by 2.2× over Araki et al.'s protocol [1] and 3.8× over ABY2 [42]. Our protocol has the highest throughput for all key sizes in WAN. Beaver et al.'s protocol [4] is not competitive in this throughput benchmark due to its high communication cost. We, hence, conjecture that it is also not competitive for large and wide circuits.

6 RELATED WORK

Two-party computation [49, 50] and multi-party computation [6, 22] have been introduced almost 40 years ago and since have been subject to intense investigation. In 2002 Maurer presented a simple multi-party computation protocol in the information-theoretic setting using replicated secret shares for educational purposes [37]. Araki et al. [1] improve this protocol by using correlated randomness from pseudo-random functions and their resulting protocol requires only one bit of communication per party for multiplication in the semi-honest setting which is optimal in the information-theoretic setting without preprocessing. Follow up work [20] extends this protocol to be secure against a malicious adversaries with increased cost of 10 bits per multiplication and an additional offline phase. Following this work, ABY3 [39] additionally designs efficient conversions between binary sharing, arithmetic sharing and Yao sharing and customized building blocks with specific focus on inference in Machine Learning (ML) models. Also tailored to ML inference, Chameleon [46] makes use of Du and Atallah's protocol [18] with the help a semi-trusted party in the offline phase which removes the requirement for all the parties to be online during the protocol execution but their communication cost per multiplication is 4 bits. Recently, ASTRA [12] was presented as a new protocol following the secret sharing scheme by Gordon et al. [23] that allows for a multiplication protocol with 2 bits of online communication. It introduces a pre-computation phase, such that the total communication cost (pre-computation plus online) equals that of Araki et al. [1]. Similar to Chameleon, only two parties have to participate during the online phase. This allows for an improvement in the running time of the protocol where each round takes the minimum latency among each pair of parties as opposed to the maximum in Araki et al.'s protocol [1]. Our protocol shares the same property with ASTRA, however, no preprocessing is needed in our protocol. Since there is no public implementation for ASTRA, we can not replicate their results. We expect the running time of ASTRA be on the same order of our protocol on circuits using only 2-fan-in gates since the communication structure of both protocols is essentially the same.

Most multi-party protocols require a number of communication rounds linear in the multiplicative depth of the circuit they compute. An exception are constant-round protocols based on Beaver et al.'s technique [4]. Beaver et al.'s original construction had a communication complexity cubic in the number of parties. Recent improvement [5, 35, 36] have communication complexity quadratic in the number of parties. However, these protocols are practically even less efficient for current circuit sizes due to their use of complex cryptography. Hence, we compare our protocol to a modified version of Beaver et al.'s original technique based on replicated secret sharing [31] and show that our protocol offers better efficiency over simulated and real WANs despite higher round complexity.

Araki et al.'s protocol and ASTRA offer near-optimal efficiency. However, their security may be criticized, because they operate in the semi-honest model. Chaudhari et al. present a version secure against malicious adversaries in the same publication [12]; Araki et al. [20] offer a separate version. There exist protocols with further efficiency improvements in the malicious model; some are three-party protocols [33, 43] and some are four party protocols [10, 13, 15]. Securing our protocol against malicious adversaries is future work.

To speed-up the online phase, Dessouky et al. [17] introduced two different 2-party semi-honest protocols that use multi-input lookup tables (LUT) rather than 2-input gates to reduce the number of communication rounds for binary circuits. The best proposed constructions for AES S-box needs 795 bytes of communication in 3 rounds and 257 bytes of communication in 1 round for both versions of their protocols, respectively. In comparison, our protocol only requires *130* bits of communication in 3 rounds which is further distributed among 3 parties. In another work, Rotaru et al. [30] extends the support for lookup tables to the multi-party setting with malicious security. By operating over binary finite fields, they improve the online communication cost of AES S-box to 48 bits in 1 round in the semi-honest 3-party case instantiated with the field $\mathbb{F}_{2^8}$. However, unlike our protocol, this requires a resource-intensive offline phase with a throughput of <1 blocks per second which hinders the applicability of the protocol in settings where preprocessing is infeasible or unwanted such as ad-hoc computations.

In addition, each evaluation round involves two-way communication between all pair of parties in contrast to our protocol. In fact, considering the real WAN setup in Section 5.3, it takes 75 ms to process a single round in Rotaru et al.'s protocol whereas ours is only 25 ms. Therefore, despite the fact that our AES S-box has 3× the number of communication rounds, the overall latency is effectively the same for both protocols in this environment. This applies to many computations over many WAN due to the difference between the maximum and minimum round trip time among the parties. We remark though that the [30] is secure in a dis-honest majority setting while our protocol works for honest-majority.

In a similar direction to ours, both Ohata et al. [41] and ABY2 [42] introduced 2-party protocols for multi-fan-in multiplication in the semi-honest model. Ohata et al. [41] modify the precomputation of Beaver multiplication triples [3] to support multi-fan-in gates. The online communication cost of this approach grows linearly with the fan-in of the multiplication. Using a new sharing scheme, the ABY2 [42] protocol requires a constant communication of just 2 bits for arbitrary fan-in multiplications. On the downside, both protocols suffer from function-dependent preprocessing with computation and communication costs that scale exponentially with the multiplication fan-in.

7 CONCLUSION

In this work, we present a new honest-majority 3-party computation protocol secure against semi-honest adversaries. We show that our protocol maintains the near-optimal communication complexity to compute 2-fan-in AND gates while it is also capable of evaluating multi-fan-in AND gates in a single round of communication. Yet, it does not require any preprocessing. We build new depth-optimized circuits for basic operations such as addition, multiplication and comparison by taking advantage of multi-fan-in AND gates. We demonstrate the performance gains of our protocol for these circuits by extensive experiments. For a common benchmark, we use the AES circuit that we further optimize with multi-fan-in AND gates. Our protocol achieves a 2-4× improvement for both latency and throughput over state-of-the-art protocols in the WAN environment.

We point out a few open problems to explore in future work. First, with the increasing demand for privacy-preserving Machine Learning (PPML), it is interesting to analyze the efficiency of our protocol in these applications. Second, our protocol is secure for an honest-majority in the semi-honest model. Achieving security against other adversary structures with malicious or covert participants is future work. The last but not the least open problem is expanding our techniques to compute multi-fan-in AND gates in a single communication round for the general n-party setting.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council (NSERC) for grants RGPIN-05849, and IRC-537591, the Royal Bank of Canada and Amazon Web Services Canada.

REFERENCES

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *Proceedings of The 23rd ACM Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016.*

[2] Shahzad Asif and Yinan Kong. 2015. Design of an Algorithmic Wallace Multiplier using High Speed Counters. In *Proceedings of the 10th International Conference of Computing for Engineering and Sciences, Istanbul, ICCES 2015, Turkey. July, 29-31, 2015.*

[3] Donald Beaver. 1991. Efficient multiparty protocols using circuit randomization. In *Proceedings of the 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991.*

[4] Donald Beaver, Silvio Micali, and Phillip Rogaway. 1990. The round complexity of secure protocols (extended Abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA.*

[5] Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. 2017. Efficient Scalable Constant-Round MPC via Garbled Circuits. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II.*

[6] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA.*

[7] Peter Bogetoft, Dan Lund Christensen, Ivan Damgard, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. 2019. Secure Multiparty Computation Goes Live. In *13th International Conference on Financial Cryptography and Data Security, FC 2009, Accra Beach, Barbados, February 23-26, 2009.*

[8] Joan Boyar, Philip Matthews, and René Peralta. 2013. Logic minimization techniques with applications to cryptology. *J. Cryptol.* 26, 2 (2013).

[9] Luís T. A. N. Brandão, Nicolas Christin, George Danezis, and Anonymous. 2015. Toward mending two nation-scale brokered identification systems. In *Proceedings of The 15th Privacy Enhancing Technologies Symposium, Philadelphia, PA, USA, June 30 – July 2, 2015.*

[10] Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. 2020. FLASH: Fast and robust framework for privacy-preserving machine learning. In *Proceedings of the 20th Annual Privacy Enhancing Technologies*, Vol. 2. 459–480.

[11] Niklas Büscher and Stefan Katzenbeisser. 2017. *Compilation for Secure Multi-party Computation.* Springer.

[12] Harsh Chaudhari, A. Choudhury, Ashish Patra, and Ajith Suresh. 2019. ASTRA: High Throughput 3PC over Rings with Application to Secure Prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop, CCSW@CCS 2019, London, UK, November 11, 2019.*

[13] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. 2020. Trident: Efficient 4PC Framework for Privacy Preserving Machine Learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020.*

[14] J. Daemen and V. Rijmen. 2002. *The Design of Rijndael: AES — The Advanced Encryption Standard.* Springer.

[15] Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. 2021. Fantastic Four: Honest-majority four-party secure computation with malicious security. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021.*

[16] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty Computation from Somewhat Homomorphic Encryption. In *Proceedings 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012.*

[17] Ghada Dessouky, Farinaz Koushanfar, Ahmad-Reza Sadeghi, Thomas Schneider, Shaza Zeitouni, and Michael Zohner. 2017. Pushing the communication barrier in secure computation using lookup tables. In *The 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017.*

[18] Wenliang Du and Mikhail J. Atallah. 2001. *Protocols for secure remote database access with approximate matching.* Springer.

[19] Marc Fischlin. 2001. A Cost-Effective Pay-Per-Multiplication Comparison Method for Millionaires. In *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings.*

[20] Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. 2017. High-Throughput Secure Three-Party Computation for Malicious Adversaries and an Honest Majority. In *Proceedings of the 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2017, Paris, France, April 30 - May 4, 2017.*

[21] Oded Goldreich. 2002. Secure Multi-Party Computation. https://www.wisdom.weizmann.ac.il/~oded/PSX/prot.pdf.

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA.*

[23] S. Dov Gordon, Samuel Ranellucci, and Xiao Wang. 2018. Secure computation with low communication from cross-checking. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III.*

[24] Vipul Goyal and Yifan Song. 2020. Malicious Security Comes Free in Honest-Majority MPC. In *ryptology ePrint Archive, Report 2020/134.*

[25] David Harris. 2003. A Taxonomy of Parallel Prefix Networks. In *Proceedings of The 37th Asilomar Conference on Signals, Systems & Computers, Pacific Grove, USA, November 9-12, 2003.*

[26] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. 2011. Faster secure two-party computation using garbled circuits. In *20th USENIX Security Symposium, USENIX Security 2011, San Francisco, CA, USA, August 8-12, 2011.*

[27] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. 2019. PILOT: Practical Privacy-Preserving Indoor Localization Using OuTsourcing. In *4th IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019.*

[28] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. 2020. On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020.*

[29] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020.*

[30] Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Peter Scholl, Eduardo Soria-Vazquez, and Srinivas Vivek. 2017. Faster Secure Multi-Party Computation of AES and DES Using Lookup Tables. In *Proceedings of The 15th International Conference on Applied Cryptography and Network Security, ACNS 2017, Kanazawa, Japan, July10-12, 2017.*

[31] Marcell Keller and Avishay Yanai. 2018. Efficient Maliciously Secure Multiparty Computation for RAM. In *Proceedings of the 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2018, Tel Aviv, Israel, April 29 - May 3, 2018.*

[32] Brian Knott, Shobha Venkataraman, Awni Y. Hannun, Shubho Sengupta, Mark Ibrahim, and Laurens van der Maaten. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual.*

[33] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021.*

[34] Sven Laur, Rilvo Talviste, and Jan Willemson. 2013. AES block cipher implementation and secure database join on the SHAREMIND secure multi-party computation framework. In *Proceedings of the 11th International Conference on Applied Cryptography and Network Security, ACNS 2013, Banff, Canada, June 25-28, 2013.*

[35] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. 2019. Efficient Constant-Round Multi-party Computation Combining BMR and SPDZ. *Journal of Cryptology* 32, 3 (2019), 1026–1069.

[36] Yehuda Lindell, Nigel P. Smart, and Eduardo Soria-Vazquez. 2016. More Efficient Constant-Round Multi-party Computation from BMR and SHE. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I.*

[37] Ueli M. Maurer. 2020. Secure Multi-party Computation Made Simple. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers.*

[38] Payman Mohassel and Peter Rindal. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *In 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017.*

[39] Peyman Mohassel and Peter Rindal. 2021. ABY3: A Mixed Protocol Framework for Machine Learning. In *Proceedings of the 25th Annual (ACM) Conference on Computer and Communications Security, Toronto, Canada, October 15-19, 2018.*

[40] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. 2012. A New Approach to Practical Active-Secure Two-Party Computation. In *Proceedings 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012.*

[41] Satsuya Ohata and Koji Nuida. 2020. Communication-Efficient (Client-Aided) Secure Two-Party Protocols and Its Application. In *24th International Conference on Financial Cryptography and Data Security, Kota Kinabalu, Malaysia, February 10–14, 2020.*

[42] Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. 2021. ABY2.0: Improved Mixed-Protocol Secure Two-Party Computation. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021.*

[43] Arpita Patra and Ajith Suresh. 2020. BLAZE: Blazing Fast Privacy-Preserving Machine Learning. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020.*

[44] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. 2009. Secure Two-Party Computation is Practical. In *Advances in Cryptology - ASIACRYPT 2009 - 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009.*

[45] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yana. 2019. "Efficient circuit- based PSI with linear communication. In *38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Eurocrypt, Darmstadt, Germany, May 19-23, 2019*.

[46] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications. In *Proceedings of the 13th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2018, Incheon, Korea, June 4-8, 2018*.

[47] Eleonora Testa, Mathias Soeken, Heinz Riener, Luca Amaru, and Giovanni De Micheli. 2019. A Logic Synthesis Toolbox for Reducing the Multiplicative Complexity in Logic Networks. In *Proceedings of the 56th Annual Design Automation Conference, Las Vegas, NV, USA, June 3-5, 2019*.

[48] C.S. Wallace. 1964. A suggestion for a fast multiplier. *IEEE Trans. Comput.* 13 (1964), 14–17.

[49] Andrew Chi-Chih Yao. 1982. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, November 3-5, 1982*.

[50] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, October 27-29, 1986*.

[51] Moti Yung. 2015. From Mental Poker to Core Business: Why and How to Deploy Secure Computation Protocols?. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*.